Prof. Dr. Matthias O. Franz

Übung 3: Naiver Bayesklassifikator

Praktikum Maschinelles Lernen

1. Gesichtserkennung

Implementieren Sie den Gaussian-Naïve-Bayes-Klassifikator aus der Vorlesung. Testen Sie Ihre Implementierung am Datensatz "Labeled Faces in the Wild" aus dem vorangegangenen Arbeitsblatt, wiederum nur für Personen, für die mindestens 70 Bilder existieren. Teilen Sie Ihren Datensatz in 60 % Trainings- und 40% Testdaten (nach vorheriger Zufalls-Permutation der Reihenfolge) und skalieren Sie die Bilder wieder auf 1/8 der Originalgröße. Führen Sie anschließend eine Hauptkomponentenanalyse auf den Trainingsdaten durch und projizieren Sie sowohl Trainings- als auch Testbilder auf die ersten 7 Eigengesichter. Trainieren Sie Ihren GNB-Klassifikator auf dem Trainingsdatensatz als "George-W.-Bush-Detektor", d.h. alle zu dieser Person gehörigen Bilder werden mit 1 gelabelt, alle sonstigen mit –1. Werten Sie Ihren Klassifikator sowohl auf den Trainings- wie auf den unabhängigen Testdaten aus. Bestimmen Sie dafür jeweils die Detektionswahrscheinlichkeit, Richtig-Negativ-Rate, Fehlalarmrate und Falsch-Negativ-Rate.

2. Textklassifikation

In dieser Aufgabe geht es um die Klassifikation von Newsgroup-Beiträgen in verschiedene Themen mithilfe einer weiteren Variante des naiven Bayes-Klassifikators, dem *multinomialen naiven Bayesklassifikator*. Der Grund für die Wahl dieses Klassifikators liegt in dem besonderen Merkmalsraum, in dem hier die einzelnen Texte dargestellt werden: für jeden Text wird die Häufigkeit des Auftretens aller Wörter gezählt und in einem Merkmalsvektor gespeichert, der so viele Einträge hat, wie es Worte im Gesamtvokabular aller vorkommenden Texte gibt. Die Likelihood für das Auftreten des Wortes x_j in Texten der Klasse ω_i modelliert man nicht durch eine Gaußverteilung wie bei *Gaussian Naive Bayes* (GNB), sondern durch eine feste Wahrscheinlichkeit p_{ij}, d.h.

$$p(x_j|\omega_i) = p_{ij} = \text{const.}$$

Die Summe der Wahrscheinlichkeiten über das gesamte Vokabular V muss für eine Klasse insgesamt 1 ergeben:

$$\sum_{x_j \in V} p(x_j|\omega_i) = \sum_{x_j \in V} p_{ij} = 1$$

Die Annahme des naiven Bayesklassifikators, dass die Likelihoods der einzelnen Merkmale voneinander unabhängig sind, bedeutet hier

$$p(\mathbf{x}|\omega_i) = p(x_1|\omega_i) \cdot p(x_2|\omega_i) \cdots = p_{i1} \cdot p_{i2} \cdots .$$

Eine solche Verteilung heisst *Multinomialverteilung*, daher der Name dieses Klassifikators.

Im Gegensatz zu GNB muss hier also pro Merkmal und Klasse nur ein Parameter geschätzt werden, nämlich p_{ij}. Eine einfache Möglichkeit zur Schätzung wäre die relative Häufigkeit des Wortes x_j in allen Texten der Klasse ω_i innerhalb eines Trainingsdatensatzes

$$\hat{p}_{ij} = \frac{n_{ij}}{N_i},$$

wobei n_{ij} die Anzahl der Vorkommnisse des Wortes x_j und N_i die Gesamtzahl aller Worte in den Trainingstexten der Klasse ω_i ist. Leider hat diese Schätzmethode den Nachteil, dass, sobald ein Wort überhaupt nicht in einer Klasse vorkommt (d.h.$\hat{p}_{ij} = 0$), die Gesamtlikelihood der Klasse gleich 0 wird, was nicht sehr realistisch ist. Man behilft sich daher mit der sogenannten Laplace-Glättung, bei der man einfach 1 zu n_{ij} hinzuzählt:

$$\hat{p}_{ij} = \frac{n_{ij} + 1}{N_i + b} \quad \text{mit} \quad b = |V|.$$

Weitere Details zum Training und zur Implementierung dieses Klassifikators lesen Sie bitte unter `http://nlp.stanford.edu/IR-book/html/htmledition/naive-bayes-text-classification-1.html` nach.

Aufgaben:

a. Die Textdatenbank *20 Newsgroups* findet sich zum Download unter `http://qwone.com/~jason/20Newsgroups/20news-18828.tar.gz` . Benutzen Sie zum Entpacken das Python-Standardpaket `tarfile`. Die Datenbank enthält ca. 20.000 Textdokumente, etwa gleichmäßig aufgeteilt auf 20 Newsgroups zu unterschiedlichen Diskussionsthemen. Die Dokumente zu jeder Newsgroup befinden sich nach dem Entpacken in einem Verzeichnis gleichen Namens.

b. Wir beschränken uns auf die vier Newsgroups `alt.atheism`, `comp.graphics`, `sci.space` und `talk.religion.misc`. Lesen Sie alle Dateien aus diesen vier Verzeichnissen in eine Array von Strings ein (d.h. ein File in einem String). Speichern Sie zusätzlich die Klassenzugehörigkeit jedes Dokuments in einem Vektor ab (Kontrolle: Sie müssten jetzt 3387 Strings im Speicher haben). Ungünstigerweise enthalten diese Dokumente immer noch den Namen der Newsgroup im Header, was eine Klassifikation trivial machen würde. Wir müssen also die ersten 2 Zeilen aus jedem Dokument entfernen, was mit folgendem Code geschieht (die Dokumente befinden sich als Strings im Array `data`):

```
def strip_header(text):
    _before, _blankline, after = text.partition('\n\n')
    return after

data = [strip_header(text) for text in data]
```

c. Im nächsten Schritt muss jeder String in Worte (*Tokens*) zerlegt werden, die durch Leerzeichen, Kommas etc. voneinander getrennt sind. Hierzu setzen wir das Python-Standardpaket `re` ein, das zur Analyse regulärer Ausdrücke dient. Durch folgenden Befehl werden alle Tokens eines Strings `textline` in einer Liste `l` gespeichert, nachdem er zuvor mit `lower()` in Kleinbuchstaben umgewandelt wurde:

```
import re
l = re.compile(r"(?u)\b\w\w+\b").findall(txtline.lower())
```

Wer sich mit regulären Ausdrücken auskennt, kann hier mit anderen Suchmustern experimentieren. Bestimmen Sie auf diese Weise die Größe des gemeinsamen Vokabulars aller Texte (Kontrolle: mit obigem Suchmuster müssten Sie 41777 unterschiedliche Tokens erhalten, d.h. Ihre Merkmalsvektoren sind daher 41777-dimensional). Berechnen Sie für jeden Text einen Merkmalsvektor, der für jedes Wort des Vokabulars seine Häufigkeit innerhalb des Texts enthält.

d. Verwenden Sie die ersten 60% der Daten als Trainingsdatensatz, die restlichen als Testdatensatz. Trainieren Sie damit einen multinomialen naiven Bayes-Klassifikator. Bestimmen Sie den Anteil korrekter Klassifikationen auf Ihren Trainings- und Testdaten. Wie gut generalisiert Ihr Klassifikator?